A Novel Fast Hybrid Cryptographic System: MARS4

Sheena Mathew, and K. Paulose Jacob

Abstract— **A novel and fast technique for cryptographic applications is designed and developed using the symmetric key algorithm "MAJE4" and the popular asymmetric key algorithm "RSA". The MAJE4 algorithm is used for encryption / decryption of files since it is much faster and occupies less memory than RSA. The RSA algorithm is used to solve the problem of key exchange as well as to accomplish scalability and message authentication. The focus is to develop a new hybrid system called MARS4 by combining the two cryptographic methods with an aim to get the advantages of both. The performance evaluation of MARS4 is done in comparison with MAJE4 and RSA.**

***Index Terms*— Cryptography, Decryption, Encryption, Message Authentication, Symmetric key.**

I. INTRODUCTION

SYMMETRIC key cryptographic algorithms, which use the same key for encryption and decryption are faster and efficient, but they have the disadvantage of key exchange problems[1]-[2]. Whereas asymmetric key cryptographic algorithms solve the major problem of key exchange as well as scalability and also achieve the purpose of non-repudiation [3]-[5]. The dawn of asymmetric key cryptography does not indicate the end of secret key cryptography. In practice, the symmetric key and asymmetric key systems are not in competition. Most cryptographic schemes on which e-commerce operations rely use a hybrid of these systems. Here the asymmetric key system is used for the distribution of a secret key, which can be a long-term key or specific to a particular communication session. Then the securely distributed secret key is used to encrypt and decrypt messages in a communication channel between two users. The performance of secret key cryptography over that of asymmetric key, and the appeal of key distribution inherent to asymmetric key cryptography, are the main reasons for the wide adoption of these hybrid systems [6].

As shown in Fig. 1 the plain text is encrypted with fast symmetric encryption algorithm, MAJE4 to form the cipher text and then the symmetric key K1 of MAJE4 is encrypted with public key K2 of asymmetric algorithm RSA [7]-[13]. Then the cipher text and encrypted symmetric key K1 are sent together by the sender to the receiver. In the receiver side, first RSA algorithm is run with its private key K3 to recover the symmetric key K1. Then by using K1 and MAJE4 the entire cipher text is converted into plain text.

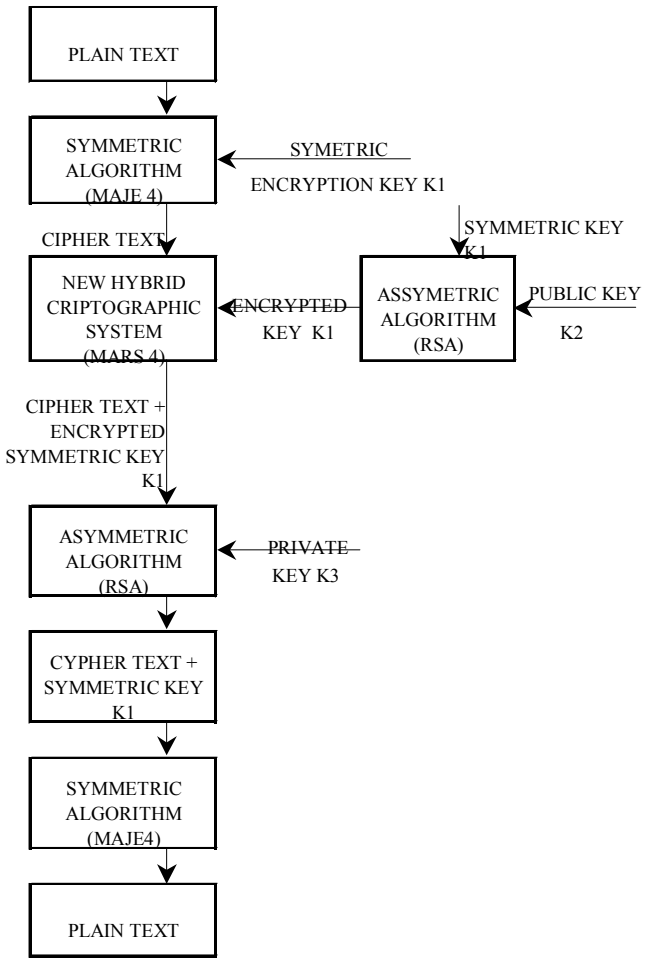

Fig. 1. Hybrid cryptographic system MARS4 using MAJE4 symmetric algorithm and RSA asymmetric algorithm.

II. DESIGN OBJECTIVES FOR MARS4

The following objectives are considered while combining the two cryptographic algorithms with a view to obtain the merits of both the systems.

1) The method should be completely secure.

Sheena Mathew is a Reader in the Computer Science and Engineering Division of Cochin University of Science and Technology, Kochi, Kerala, India (e-mail: sheenamathew@cusat.ac.in).

K. Paulose Jacob is a Professor and Dean of Faculty of Engineering of Cochin University of Science and Technology, Kochi, Kerala, India (e-mail: kpj@cusat.ac.in).

1-4244-0370-7/06/$20.00 ©2006 IEEE

2) The encryption / decryption process should not take longer time.
3) The generated cipher text should be compact in size.
4) The solution should scale to a large number of users easily, without introducing any additional complications.
5) The key exchange problem should be solved by the new method.

III. DESCRIPTION OF MARS4

All the above-mentioned design considerations were taken care while designing MARS4. The following are the main features of MAJE4, RSA and MARS4.

A. MAJE4

1) Main features of MAJE4

1) The encryption sequence can have a large period.
2) The key stream can approximate the properties of a true random stream.
3) MAJE4 is suitable for hardware or software and it uses only primitive computational operations commonly found in microprocessors.
4) It is simple and fast. It uses simple algorithm, which is easy to implement and eases the task of determining the strength of the algorithm.
5) Low memory requirement makes it suitable for handheld type devices with restricted memory.
6) Mixed operators are used for the design of MAJE4. The use of more than one arithmetic and / or Boolean operator complicates cryptanalysis. Primitive operators like + and ^ are used since these operators do not commute and hence cryptanalysis becomes again more difficult.
7) Variable number of rounds is used. An increase in the number of rounds increases cryptanalytic strength.

2) Key setup of MAJE4

One can choose either a 128-bit key or a 256-bit key.

128-bit key: The first four 32 bit words, ie. $key_{[0]}$, $key_{[1]}$, $key_{[2]}$ and $key_{[3]}$ are considered for storing the key.

256-bit key: The key is stored in eight 32 bit words $key_{[0]}$, $key_{[1]}$, $key_{[2]}$, $key_{[3]}$, $key_{[4]}$, $key_{[5]}$, $key_{[6]}$ and $key_{[7]}$.

3) Algorithm of MAJE4

Steps:

1) Assign the key length kl either as 128-bit or 256-bit.
2) if kl = 128 then div=4
   else div=8
3) if kl = 128 then consider two lsb's of $key_{[0]}$, find the decimal equivalent of these two lsb's and store in the variable 'in'.
   else
   if kl = 256 then consider three lsb's of $Key_{[0]}$, find the decimal equivalent of these three lsb's and store it in a variable 'in'.
4) ran =$key_{[0]}$ ^ $key_{[in]}$
5) if kl = 128 then consider two lsb's of ran, find the decimal equivalent of that and store in the variable 'in1'.
6) if kl = 256 then consider three lsb's of ran, find the decimal equivalent of that and store in the variable 'in1'.
7) check the 16^{th} bit in ran,
   if it is 1 then
   newran=($key_{[in1]}$ + $key_{[in1+1 mod div]}$) ^ ($key_{[in1+2 mod div]}$ + $key_{[in1+3 mod div]}$)
   else
   newran=($key_{[in1]}$ ^ $key_{[in1+1 mod div]}$) + ($key_{[in1+2 mod div]}$ ^ $key_{[in1+3 mod div]}$)
8) The output 32-bit word is newran, which can be used to XOR with the corresponding word in the plain text
9) Advance all the keys as
   $key_{[i]} = key_{[i]} * key_{[i]} + key_{[i]} >>20$
10) go to step3

B. RSA

1) Main features of RSA

1) RSA is computationally easy for a party B to generate the key pair (Public key KS_b, Private key KR_b).
2) It is computationally easy for a sender A, knowing the public key KS_b and the message to be encrypted M, to generate the cipher text $C=E_{KSb}(M)$.
3) It is computationally easy for the receiver B, to decrypt the resulting cipher text using the private key to recover the original message $M=D_{KRb}(C)=D_{KRb}[E_{KSb}(M)]$
4) It is computationally infeasible for an opponent, knowing the public key KS_b alone to determine the private key KR_b.
5) It is also computationally infeasible for an opponent, knowing the public key KS_b, and a cipher text C, to recover the original message M.
6) The encryption and decryption functions can be applied in either order. $M=D_{KRb}[E_{KSb}(M)]= D_{KSb}[E_{KRb}(M)]$

2) Algorithm of RSA

1) Choose two large prime numbers P and Q.
2) Calculate N=P*Q.
3) Select the public key (encryption key) E such that it is not a factor of (P-1) and (Q-1).
4) Select the private key (decryption key) D such that the following equation is true:
   (D*E) mod (P-1) * (Q-1) = 1
5) Encrypt the plain text PT to form the cipher text CT as follows
   $CT = PT^E \bmod N$
6) Send CT as the cipher text to the receiver.

7) Decrypt the cipher text CT to form the plain text PT as follows
$PT = CT^D \bmod N$

The crux of RSA is that factoring N to find P and Q is not at all easy and is quite complex and time consuming.

C. MARS4

Now MAJE4 and RSA can be combined to have MARS4 as a very efficient security solution. Assume that A is the sender of a message and B is the receiver. MARS4 is designed to work as follows.

1) A encrypts the original plain text message (PT) with the help of MAJE4 and the symmetric key (K1) and forms the cipher text (CT).
2) Encrypt K1 with the public key (K2) of B using RSA.
3) Attach the encrypted K1 to the CT and send it to B.
4) B now uses the RSA algorithm and its private key (K3) to decrypt K1.
5) Then B uses K1 and the MAJE4 algorithm to decrypt the CT for yielding the original plain text (PT).

As specified in the design objectives of MARS4, the symmetric key algorithm MAJE4 is faster and it can produce 352 Mbps. Also the generated cipher text is of the same size as the plain text. Instead, if we had used the asymmetric key encryption as in RSA, then the operation would have been quite slow, especially when N is large as shown in Table IV. Also the cipher text produced is of larger size than the size of the plain text. Now since the encryption of only 128 bit private key is done using RSA, the encryption process will not take long time and the encrypted key will not consume more memory space. Thus RSA is used for solving the major problem of key exchange. MARS4 is thus having the advantages of both MAJE4 and RSA.

IV. RESULTS

Tables I to V show the results of the MAJE4, RSA and MARS4 when run with plain text of different sizes. The memory sizes of the plain text to be encrypted, the cipher text and also the time taken for encryption and decryption are also given.

TABLE I
TIME TAKEN FOR ENCRYPTION OR DECRYPTION OF FILES OF VARIOUS SIZES USING MAJE4

File size of plain text (bytes)	File size of cipher text (bytes)	Time taken (Sec.)		
		Encryption	Decryption	Total
30144	30144	0.01	0.01	0.02
60003	60003	0.02	0.02	0.04
90070	90070	0.03	0.03	0.06
120014	120014	0.04	0.04	0.08
150060	150060	0.05	0.05	0.10

TABLE II
TIME TAKEN FOR ENCRYPTION OR DECRYPTION OF FILES OF VARIOUS SIZES USING RSA (N=187)

File size of plain text (bytes)	File size of cipher text (bytes)	Time taken (Sec.)		
		Encryption	Decryption	Total
30144	101336	0.03	0.04	0.07
60003	201672	0.06	0.09	0.15
90070	302665	0.09	0.14	0.23
120014	403183	0.12	0.19	0.31
150060	504041	0.15	0.24	0.39

TABLE III
TIME TAKEN FOR ENCRYPTION OR DECRYPTION OF FILES OF VARIOUS SIZES USING RSA (N=3431)

File size of plain text (bytes)	File size of cipher text (bytes)	Time taken (Sec.)		
		Encryption	Decryption	Total
30144	140080	0.05	0.07	0.12
60003	278891	0.10	0.14	0.24
90070	418769	0.15	0.22	0.37
120014	557735	0.20	0.29	0.49
150060	697224	0.25	0.36	0.61

TABLE IV
TIME TAKEN FOR ENCRYPTION OR DECRYPTION OF FILES OF VARIOUS SIZES USING RSA (N=44377)

File size of plain text (bytes)	File size of cipher text (bytes)	Time taken (Sec.)		
		Encryption	Decryption	Total
30144	166888	0.03	0.09	0.12
60003	332224	0.06	0.18	0.24
90070	498816	0.09	0.27	0.36
120014	664584	0.12	0.36	0.48
150060	830967	0.15	0.46	0.61

TABLE V
TIME TAKEN FOR ENCRYPTION OR DECRYPTION OF FILES OF VARIOUS SIZES USING MARS4

File size of plain text (bytes)	File size of cipher text + key (bytes)	Time taken (Sec.)		
		Encryption	Decryption	Total
30144	30160	0.01	0.01	0.02
60003	60019	0.02	0.02	0.04
90070	90086	0.03	0.03	0.06
120014	120030	0.04	0.04	0.08
150060	150076	0.05	0.05	0.10

V. PERFORMANCE EVALUATION

The performance evaluation is done by comparing the time taken and memory space required for encryption and decryption using MAJE4, RSA and MARS4 algorithms. The evaluation is done using Pentium IV processor, Linux operating system and C compiler.

A. Timing Analysis

As shown in Table VI, the time taken for the MAJE4 symmetric crypto system and the MARS4 hybrid crypto system can be seen as the same. Hence the advantage of symmetric system, which is the speed of encryption and decryption, is preserved in the hybrid system also. Whereas RSA 8 bit algorithm is taking 0.07 sec., RSA 12bit and 16 bit are taking 0.12 sec. each. RSA12 bit and RSA 16 bit are taking the same time since the data is processed byte by byte. There is a difference of 0.05 sec. when RSA 8 bit and RSA 16 bit are considered. If we consider this difference and calculate the time required for RSA 128 bit, then the time required is obtained as 0.82 sec. Hence the MARS4 is 41 times faster than RSA. Since the RSA is used in MARS4, the key exchange problem of MAJE4 is also solved.

TABLE VI
TIME REQUIRED FOR ENCRYPTION AND DECRYPTION USING MAJE4, RSA, AND MARS4.

Algorithm used	Key length	File size of plain text (bytes)	Time taken (sec.)
MAJE4	128 bit	30144	0.02
RSA	8 bit	30144	0.07
RSA	12 bit	30144	0.12
RSA	16 bit	30144	0.12
MARS4	128 bit	30144	0.02

B. Memory Requirements

When the memory requirements for the cipher text of MAJE4 and MARS4 are compared as shown in Table VII, both are found taking almost the same memory size. In RSA 8 bit the memory requirement is almost 4 times greater than that of MARS4 and in RSA 12 bit, it is 5 times greater than that of MARS4. Also RSA16 bit is 6 times greater in size than MARS4. Hence for each additional 4-bit key in RSA, the memory size can be found to be increasing by about the memory size of the given plain text as shown in Fig. 2

TABLE VII
MEMORY REQUIREMENTS FOR MAJE4, RSA AND MARS4

Algorithm used	Key length	File size of plain text (bytes)	File size of cipher text (bytes)
MAJE4	128bit	30144	30144
RSA	8bit	30144	101336
RSA	12bit	30144	140080
RSA	16bit	30144	166888
MARS4	128bit	30144	30160

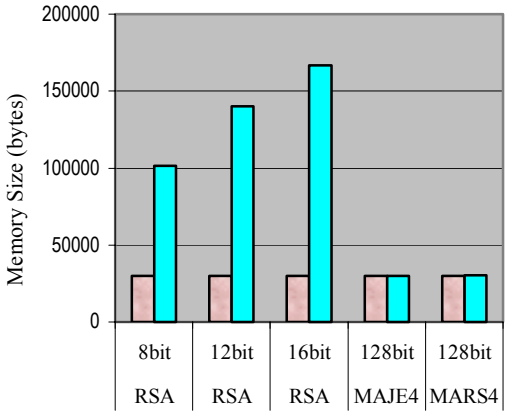

Fig. 2. Comparison of memory sizes in bytes for MAJE4, RSA and MARS4.

Thus if RSA 128 bit key is used, the memory size will be about 34 times greater than that of MARS4.

VI. CONCLUSION

From the analysis of results and performance evaluation, it can be concluded that MARS4 can be used as a reliable hybrid crypto system, which is much faster than the popular RSA. The memory requirement for MARS4 is also less than RSA. The key exchange problem in MAJE4 can be solved by using MARS4.

Thus the advantages of both cryptographic systems are preserved in the hybrid system MARS4. Because of the low memory requirement, it can be used in handheld devices with restricted memory. Since it is faster, it can be used for applications that require encryption / decryption of a stream of data sent through the Internet. Message authentication can also be achieved with the help of RSA used in MARS4. Hence this MARS4 hybrid system proves to be a very sound technique for transferring messages from sender to the receiver, achieving confidentiality as well as message authentication.

VII. REFERENCES

[1] Kencheng Zeng, Chung-Huang Yang, Dah-Yea Wei and T.R.N. Rao, "Pseudorandom Bit Generators in Stream-Cipher Cryptography", *IEEE Computer*, February 1991, pp 8-17

[2] Mustak E. Yalcin, Johan A. K. Suykens and Joos Vandewalle, "True Random Bit Generation From a Double-Scroll Attractor", *IEEE Transactions on Circuits and Systems*, Vol. 51, No.7, July 2004, pp 1395 –1404

[3] Fujisaki E. and Okamoto T., "How to Enhance the Seurity of Public-Key Encryption at Minimum Cost", Proc. Of PKC' 99 , Springer-Verlag, LNCS 1560,1999, pp.53-68.

[4] H. C. Williams, " A Modification of the RSA Public key Encryption Procedure", IEEE Trans. On Information Theory, Vol. IT-26, No.6, 1980, pp.726-729.

[5] Bellare M. and Rogaway P," Optimal Asymmetric Encryption", Proc. Of Eurocrypt'94,LNCS 950, Springer-Verlag 1995, pp.92-111.

[6] Fujisaki E. and Okamoto T," Secure Integration of Asymmetric and Symmetric Encryption Schemes", Proc. Of Crypto'99, Springer – Verlag LNCS 1666, 1999, pp 535-554.

[7] Sheena Mathew, K. Paulose Jacob "Performance Evaluation of Pseudo Random Number Generators – A Statistical Analysis", Proceedings of International Conference on Resource Utilisation and Intelligent Systems (INCRUIS) -2006, pp 224-229

[8] J. Boyar, "Inferring Sequences Produced by Pseudo-Random Number Generators", *Journal of ACM*, Vol.36 (1), Jan 1989, pp 129-141.

[9] Park Stephen K. and Keith W. Miller, "Random Number Generators: Good ones are hard to find", *Communications of the ACM*, October 1988, pp.1192-1201.

[10] William Aiello, Sivaramakrishnan Rajagopalan and Ramarathnam Venkatesan, "Design of Practical and Provably Good Random Number Generators", *SODA'95, ACM* Jan 1995.pp 1-9.

[11] T. Seigenthaler, "Decrypting a class of Stream Ciphers Using Cipher text Only" , *IEEE Transactions on computer*, Vol C-34, No.1, Jan 1985, pp 81-85.

[12] Herve Chabanne and Emmanuel Michon, "Jeroboam", *5th International Workshop on Fast Software Encryption*, Springer-Verlag, London, 1998, pp 49-59.

[13] Sheena Mathew, K. Paulose Jacob "A New Fast Stream Cipher: MAJE4", Proceedings of IEEE, INDICON 2005, pp 60-63.